COMPLETE LISTING OF THE CLAIMS

1. (Currently Amended): A system that facilitates communicating between managed and unmanaged code, comprising:

a first component that is ~~one of the~~ managed ~~and unmanaged~~ code; and

a caller associated with the first component, the caller ~~having code that~~ invokes an object related to a second component, the second component being ~~one of the managed and~~ unmanaged code, the caller's code ~~including~~ includes an in-lined stub that facilitates communications to ~~between the objects~~ and execution management of the object.

2. (Currently Amended): The system of claim 1, the in-lined stub including a call and return pair ~~to~~ that facilitate the communications and execution management of the object ~~between the objects~~.

3. (Currently Amended): The system of claim 1, further comprising a stack marker that is hoisted from within a code loop associated with the caller ~~to~~ that facilitates code execution performance during communications with the object ~~between the objects~~.

4. (Currently Amended): The system of claim 1, the caller further comprising transition code ~~to~~ that synchronizes execution between the objects.

5. (Currently Amended): The system of claim 1, the caller further comprising one or more flags ~~to~~ that synchronize execution of the object ~~between the objects~~.

6. (Original): The system of claim 5, the one or more flags utilized to synchronize code execution with a garbage collector.

7. (Currently Amended): The system of claim 6, the one or more flags ~~utilized to~~ suspend return operations from the unmanaged code until operations associated with the garbage collector have completed.

8. (Currently Amended): The system of claim 1, the caller further comprising security attribute code ~~to~~ that insulates the ~~objects~~ unmanaged code from at least one of code and security implementation details.

9. (Original): The system of claim 1, the caller further comprising calling convention code to at least one of organize arguments and an execution stack according to the convention of the unmanaged code.

10. (Original): The system of claim 9, the calling convention code utilized to interpret one or more return values from the unmanaged code.

11. (Currently Amended): The system of claim 1, the caller including at least one of an in-lined marshalling code and an external marshalling code ~~to~~ that transfers data between the ~~objects~~ managed and the unmanaged code.

12. (Currently Amended): The system of claim 1, the caller further comprising an extensibility component ~~to~~ that facilitates generalized communications between the ~~objects~~ managed and the unmanaged code.

13. (Original): The system of claim 12, the extensibility component including a function pointer that includes one or more functions as arguments.

14. (Currently Amended): A method ~~to facilitate~~ of facilitating communications between managed and unmanaged code, comprising:

incorporating external stub code functionality in-line within code of a calling function, the external stub code functionality includes communication and execution management functions between managed and unmanaged code; and

providing a call and return within the calling code ~~to~~ that facilitates the communication~~s~~ and management functions between the managed and unmanaged code.

15. (Original): The method of claim 14, further comprising hoisting a marker from a loop associated with the caller to facilitate code execution performance during communications between the managed and unmanaged code.

16. (Original): The method of claim 14, further comprising synchronizing execution between the managed and unmanaged code.

17. (Original): The method of claim 16, further comprising synchronizing code execution with a garbage collector.

18. (Original): The method of claim 17, further comprising suspending operations from the unmanaged code until operations associated with the garbage collector have completed.

19. (Currently Amended): The method of claim 14, further comprising utilizing security attributes ~~to~~ that insulate the managed and unmanaged code from at least one of code and security implementation details.

20. (Original): The method of claim 14, further comprising at least one of organizing arguments and an execution stack according to the format of the unmanaged code.

21. (Original): The method of claim 14, transferring data between the managed and unmanaged code *via* at least one of an in-lined marshalling code and an external marshalling code.

22. (Currently Amended): The method of claim 14, further comprising generalizing a function ~~to~~ that facilitates communications between the managed and unmanaged code.

23. (Currently Amended): The method of claim 22, further comprising utilizing a function pointer that points to one or more functions as arguments ~~to~~ that facilitate communications between the managed and unmanaged code.

24. (Original): A computer-readable medium having computer-executable instructions for performing the acts of claim 14.

25. (Currently Amended): A system ~~to~~ that facilitates communications between disparate objects, comprising:

means for incorporating external stub code functionality within code of a caller, the external stub code functionality includes communication and execution management functions between the disparate objects; and

means for interfacing with the caller to facilitate the communications and management functions between the disparate objects.

26. (Previously Presented): The system of claim 25, further comprising means for hoisting a marker from a loop associated with the caller to facilitate code execution performance during communications between the disparate objects.

27. (Currently Amended): A computer-readable medium having stored thereon, a caller ~~having code~~ that incorporates an in-lined stub ~~to~~ that executes and manages a remote object, comprising:

a first in-lined data set providing call and return interface capabilities within the caller;

a second in-lined data set providing execution stack instructions for the caller; and

a third in-lined data set providing hoisting capabilities within the caller, the interface, the stack instructions and the hoisting capabilities enabling the execution of the remote object.

28. (Currently Amended): A computer-implemented control signal ~~to~~ that communicates between managed and unmanaged code, comprising:

a control signal ~~to~~ that communicates data between the managed and unmanaged code; and

a caller associated with the managed code, the caller ~~having code that~~ invokes an object related to the unmanaged code *via* the control signal, the caller's code ~~including~~ includes an in-lined stub that facilitates communications and execution management between the managed and the unmanaged code.

29. (Currently Amended): A system that facilitates communication, the system comprising:

a caller component that is associated to a first component, the first component being ~~one of~~ managed ~~and unmanaged~~ code;

a second component that is invoked by ~~code associated with~~ an in-lined stub within the caller component, the second component is ~~the other one of the managed and~~ unmanaged code, the ~~code includes an~~ in-lined stub ~~that~~ facilitates communication and execution management between the components.

30. (Previously Presented): The system of claim 29, the in-lined stub having a call and return pair that facilitates the communication.

31. (Previously Presented): The system of claim 29, further comprising a code loop associated with the caller that hoists a stack marker that facilitates code execution performance during the communication.

32. (Currently Amended): A method ~~to facilitate~~ of facilitating communications between managed and unmanaged code, the method comprising:

embedding an in-lined stub code within code of a calling function, the in-lined stub code includes communication and execution management functionality between the managed and unmanaged code; and

incorporating a call and return within the calling code to facilitate the communication~~s~~ and execution management functionality between the managed and unmanaged code.

33. (Previously Presented): The method of claim 32, further comprising hoisting a marker from a loop, the marker facilitates code execution performance during communications between managed and unmanaged code.

34. (Previously Presented): The method of claim 32, further comprising synchronizing execution between the managed and unmanaged code.

35. (Previously Presented): The method of claim 32, further comprising transmitting data between the managed and unmanaged code *via* at least one of an in-lined marshalling code and an external marshalling code.

36. (Currently Amended): A system ~~to~~ that facilitates communications between disparate objects, the system comprising:

means for incorporating an in-lined stub within code of a caller, the in-lined stub includes communication and execution management functionality; and

means for providing a call and return within the caller ~~to~~ that facilitates the communication and execution management functionality between the disparate objects.

37 (Previously Presented): The system of claim 36, further comprising means for hoisting a marker from a loop associated with the caller to facilitate execution performance during communications between the disparate objects.

38. (Previously Presented): The system of claim 36, further comprising means for synchronizing execution between the disparate objects.

39. (Previously Presented): The system of claim 36, further comprising means for insulating the disparate objects from at least one of code and security implementation details.

40. (Previously Presented): The system of claim 36, further comprising means for suspending return operations from one of the disparate objects associated with unmanaged code until operations associated with a garbage collector have completed.